Virtual Network Embedding based on Node Connectivity Awareness and Path Integration Evaluation

Zhiyuan Zhao, Xiangru Meng, Yuze Su and Zhentao Li
College of Information and Navigation, Air Force Engineering University
Xi'an, Shanxi710077 - China
[e-mail: zhaozhiyuan_0815@126.com]
*Corresponding author: Xiangru Meng

Received December 24, 2016; revised March 15, 2017; accepted April 24, 2017; published July 31, 2017

Abstract

As a main challenge in network virtualization, virtual network embedding problem is increasingly important and heuristic algorithms are of great interest. Aiming at the problems of poor correlation in node embedding and link embedding, long distance between adjacent virtual nodes and imbalance resource consumption of network components during embedding, we herein propose a two-stage virtual network embedding algorithm NA-PVNM. In node embedding stage, resource requirement and breadth first search algorithm are introduced to sort virtual nodes, and a node fitness function is developed to find the best substrate node. In link embedding stage, a path fitness function is developed to find the best path in which available bandwidth, CPU and path length are considered. Simulation results showed that the proposed algorithm could shorten link embedding distance, increase the acceptance ratio and revenue to cost ratio compared to previously reported algorithms. We also analyzed the impact of position constraint and substrate network attribute on algorithm performance, as well as the utilization of the substrate network resources during embedding via simulation. The results showed that, under the constraint of substrate resource distribution and virtual network requests, the critical factor of improving success ratio is to reduce resource consumption during embedding.

Keywords: Virtual Network Embedding, Node Connectivity Awareness, Path Integration Evaluation, Performance Analysis, Position constraint

This work was jointly supported by the National Natural Science Foundation of China (No. 61201209, 61401499)

https://doi.org/10.3837/tiis.2017.07.005 **ISSN : 1976-7277**

1. Introduction

Network virtualization has been regarded as a fundamental technology for the future network [1, 2], which separates the control plane from the data plane, and allows multiple virtual networks (VNs) to operate on the shared substrate network (SN) simultaneously by the mechanism of resource abstraction and isolation, each virtual network can run their own specific network architecture, protocols and services independently. Virtualization reduces the ossifying forces at work in the current internet and promotes innovative network technologies.

The virtual network embedding problem is a main challenge of network virtualization, which has attracted wide attention in recent years. VN embedding [3, 4] is to embed the nodes and links of VN request to the nodes and paths of SN on the basis of resources constraints. The problem has been proved to be NP-hard [3]. Current researches mainly focus on improving the acceptance ratio of VN requests and revenue. Lots of heuristic algorithm [5-23] and meta heuristic algorithm [24, 25] have been proposed to solve the problem by an approximate optimal solution.

VN embedding algorithms include two categories: one-stage algorithm [7, 8] and two-stage algorithm [9-25] according to embedding manner. One-stage algorithm embeds virtual nodes and links at the same time by considering the constraints of both, and makes better performance. However, synchronization of node and link embedding leads to great solution space which costs much more time. In contrast, two-stage algorithm divides embedding process to node embedding stage and link embedding stage, which embeds nodes first by satisfying the requirement of virtual nodes in a greedy way, then finds loop-free paths in substrate network meeting the needs of virtual links. By this way, solution space for each stage is restricted and the computing rate is much faster, thus two-stage algorithm has been quickly developed and many relative algorithms have been proposed.

However, separation of the two stages incurs problems: poor correlation in node embedding and link embedding leads to virtual nodes be embedded far away from each other, results in the long distance of link embedding and more resource consumption; the connectivity of virtual nodes is less considered or used by an ineffective way, which leads to long distance between adjacent virtual nodes during embedding, and results in more resource consumption; the imbalance resource consumption between network components and their neighbors in substrate network leads to bottleneck and resource fragment, as a result, the link embedding of latter VNs will be bypass and expend more resources. The problems mentioned above make the ineffective utilization of substrate network resources and the poor embedding performance.

Existing algorithms have adopted different methods to solve the problems. Virtual topology connection feature [10], network attributes of substrate networks [11], connectivity of substrate nodes [12] and so on do help improve embedding performance by introduce better correlation between the node embedding and the link embedding stages. Breadth first search (BFS) algorithm [14], virtual network attributes [15] and the like can help utilize the adjacent relationship of virtual nodes. Load balance [16] and topology awareness [17] can help reduce resource fragment and promote embedding performance. Satria et al. in [26] connect the disconnected nodes to a device of its neighbor as ad-hoc relay nodes to bridge to the edge computing network. All these strategies promote embedding in different fields and different degrees. However, some papers adopted new indicators and made a commonly method to sort virtual nodes and substrate nodes. As the big difference of scale and resources between VN

and SN, the indicators make different sense and they were integrated in an ineffective way. How these strategies affect the embedding process, which indications and strategies are the major elements, and how to integrate them effectively still need more efforts.

On the other hand, VN embedding needs to consider position constraints in some scenes, in which position of substrate nodes, position and distance constraints of virtual nodes are specific. Position constraint restricts solution space of node embedding and has an evident impact on VN embedding performance. In some papers, position constraint is considered, while the distance constraint settings are different, they take position constraint as restriction during embedding, but the impact on the performance has not been well investigated [14, 18]. Besides, the substrate network topologies used in previous works are mostly random, indicators play different roles in different network styles. Impact of substrate network attributes on the performance of embedding algorithms is seldom investigated.

In this paper, based on the previous researches, we select several indicators and integrate them in new way, and present the new two-stage virtual network embedding algorithm NA-PVNM. In node embedding stage, we further separate the node mapping stage into virtual nodes sorting process and candidate substrate nodes sorting process, and sort them by different methods. We sort virtual nodes by their resource requirement and BFS algorithm; for each virtual node to be embedded, its candidate substrate nodes are sorted by their residual resources and correlation properties with former selected nodes; in this way, we take full advantage of the connectivity in both virtual and substrate topologies, and coordinate node and link embedding stages. In link embedding stage, available bandwidth, CPU and hop counts of path are taken into account to select the best path, which help reduce the imbalance consumption of substrate network resources. Besides, motivated by the lack in investigations of position constraint and substrate network attributes, we design simulations to explore the impact of position constraint and substrate network attributes on algorithm performance, and analyze the key factors to improve algorithm performance.

The main contributions of this paper can be summarized as follows: (1) We adopt different indicators and strategies in virtual nodes sorting, candidate substrate nodes sorting process, link embedding stage, and integrate them in an effective manner. (2) We discuss the impact of position constraint and substrate network attributes on the performance by controlling the position constraint and the generating of substrate networks. (3) Extensive simulation is conducted to analyze the key factors that affect the performance of VN embedding algorithm.

The rest of paper is organized as: we discuss the related works in section 2. Section 3 gives the model of VN embedding and evaluation indicators. Section 4 presents our novel VN embedding algorithm. Section 5 describes simulation results and analysis. The paper is concluded in section 6.

2. Related Work

The VN embedding problem is NP-hard, some previous algorithms (e.g. [5, 6]) usually introduce greedy strategy to node embedding stage which helps to satisfy the resource requirements of VN request and balance the loads of substrate network, but the separate between node embedding stage and link embedding stage results in more resource consumption. Therefore, better correlation between two stages was introduced by using certain network topological attribute or facilitating the latter stage when embedding the virtual nodes to substrate networks. Feng et al. in [9] select substrate node by considering the node degree along with CPU and adjacent bandwidth of node. Cui et al. in [10] introduce the node connection-degree based on virtual topology connection feature which increases the

utilization efficiency of substrate network. Wang et al. in [11] redefine closeness to consider topology attributes with the dynamic states of the nodes and edges at the same time which achieves better performance. Ding et al. in [12] introduce betweenness centrality to sort virtual nodes, correlation properties between substrate nodes are used to coordinate node embedding and link embedding. Liao et al. in [15] consider topology attributes of substrate and virtual networks through multiple characteristics to better coordinate node and link embedding. Gong et al. in [18] consider the local and global importance of network nodes in node embedding stage, and rank virtual and substrate nodes by TOPSIS.

To make a better utilization of the connectivity between virtual nodes, the BFS algorithm is introduced to sort virtual nodes in [14, 19], which is testified to help decrease the path length in link embedding stage. Peng in [14] uses the BFS algorithm and the synchronization strategy of traversing virtual and substrate nodes, embeds virtual nodes and virtual links in a coordinated way, which reduces the path length of virtual link.

To deal with the resource fragmentation in substrate network during VN embedding, Qi et al. in [16] propose balanced link load VN construction algorithm and balanced node load VN construction algorithm, then based on them, a balanced adaptive VN construction algorithm is proposed which effectively improve the performance. Li et al. in [17] define a metric called resource fragmentation degree (RFD) to measure the status of resource fragmentation, formulate the VN embedding problem as a mixed integer program, and solve it by the optimization objective of minimizing RFD in substrate network. Beck et al. in [22] propose a resource allocation model to match nodes and their adjacency links which enables nodes embedding to adapt links resource state, and solve the mismatching of allocating network resource.

3. Virtual Network Embedding Problem

In this section, the network model and evaluation indications of VN embedding problem are formulated and elaborated.

3.1 Network Model

Substrate Network a SN is modeled as a weighted undirected graph $\boldsymbol{G}_S = (N_S, L_S, A_s^N, A_s^L)$, where N_S represents the set of substrate nodes whose number is $|N_s|$. L_S represents the set of substrate links whose number is $|L_s|$. A_s^N represents the set of substrate node attributes, for a given node $n_s \in N_s$, we take the available CPU resources and position as its attributes, which are denoted as $cpu(n_s)$ and $loc(n_s) = (x_s, y_s)$, (x_s, y_s) is the two-dimensional coordinates of n_s. A_s^L represents the set of substrate link attributes, for a given link $l_s \in L_s$,we take the available bandwidth as its attribute and denote it as $bw(l_s)$.

Virtual Network Request a VN request is modeled as a weighted undirected graph $\boldsymbol{G}_v = (N_v, L_v, A_v^N, A_v^L, T_v)$, where N_v represents the set of virtual nodes whose number is $|N_v|$. L_v represents the set of virtual links whose number is $|L_v|$. A_v^N represents the set of virtual node attributes, for a given node $n_v \in N_v$, we take the requirement of CPU, position and position constraint as its attributes and denote them by $cpu(n_v)$, $loc(n_v)$ and $D(n_v)$ respectively. A_v^L represents the set of virtual link attributes, for a given link $l_v \in L_v$, we take the requirement of bandwidth as its attribute and denote it as $bw(l_v)$. T_v represents the survival time of $\boldsymbol{G}_v$. **Table**

1 shows the briefly definitions of the commonly used notations.

Virtual Network Embedding the VN embedding is defined as an embedding action M from G_v to a subset of G_s, which can be denoted as $M: G_v \to (N_s^{sub}, L_s^{sub}, R_N, R_L)$, where $N_s^{sub} \in N_s$, $L_s^{sub} \in L_s$, R_N and R_L denote the CPU and bandwidth resource of SN that allocated to the VN request.

For two-stage embedding algorithm: the node embedding stage is denoted as $M^N: (N_v, A_v^N) \to (N_s^{sub}, R_N)$; the link embedding stage is denoted as $M^L: (L_v, A_v^L) \to (L_s^{sub}, R_L)$. **Fig. 1** shows an example of embedding a VN to SN, results of M^N and M^L are $\{1 \to A, 2 \to C, 3 \to F\}$ and $\{(1,2) \to (A,C), (1,3) \to (A,D,F)\}$.

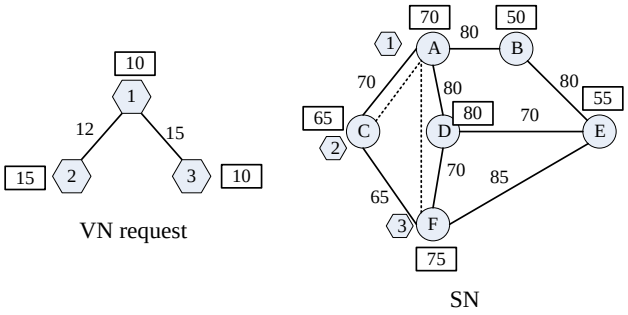


Fig. 1. An example of virtual network embedding

Table 1. Notation Table

Notations	Definitions
$\boldsymbol{G}_S$	Substrate network
N_S	the set of substrate nodes
L_S	the set of substrate links
A_s^N	the set of substrate node attributes
$cpu(n_s)$	available CPU resources of n_s
$loc(n_s)$	position of n_s
A_s^L	the set of substrate link attributes
$bw(l_s)$	available bandwidth of l_s
$\boldsymbol{G}_v$	Virtual network
N_v	the set of virtual nodes
L_v	the set of virtual links
A_v^N	the set of virtual node attributes
A_v^L	the set of virtual link attributes
$D(n_v)$	position constraint of n_v
T_v	the survival time of $\boldsymbol{G}_v$

3.2 Evaluation Indicators

The VN embedding problem is a multi-objective optimization problem under resource constraint of nodes and links, the main goal is to make full use of substrate network resources,

increase revenue and reduce cost during VN requests arriving and surviving. The performance of the proposed algorithm is evaluated by indicators including the acceptance ratio, the revenue to cost ratio, and the average hop counts of link embedding.

The acceptance ratio is defined as

$$\lim_{T\to\infty}\frac{\sum_{t=0}^{T}VN_{map}(t)}{\sum_{t=0}^{T}VN(t)} \tag{1}$$

where $VN(t)$ is the number of VN requests that arrives at time t, $VN_{map}(t)$ is the number of VN requests that have been successfully embedded at time t.

The revenue to cost ratio (R/C)

Revenue of G_v at time t is defined as

$$R(G_v,t)=\sum_{n_v\in N_V}cpu(n_v)+\alpha\sum_{l_v\in L_V}bw(l_v) \tag{2}$$

The formula shows that the revenue of G_v is the sum of resource requirement, α is the weighting coefficient to balance the relative revenues from CPU and bandwidth, we set $\alpha=1$ in this paper.

Cost of G_v at time t is defined as

$$Cost(G_v,t)=\sum_{n_v\in N_V}cpu(n_v)+\beta\sum_{l_v\in L_V}hops(l_v)\cdot bw(l_v) \tag{3}$$

The formula shows that the cost of G_v is the sum of substrate network resource consumption. $hops(l_v)$ is the total hop counts of path that l_v embed to, β is a weighting coefficient to balance the relative costs from CPU and bandwidth, we set $\beta=1$ in this paper.

So the R/C is defined as

$$R/C=\lim_{T\to\infty}\frac{\sum_{t=0}^{T}\sum_{G_v\in VN_{map}(t)}R(G_v,t)}{\sum_{t=0}^{T}\sum_{G_v\in VN_{map}(t)}Cost(G_v,t)} \tag{4}$$

Average Hop Counts of Link Embedding is defined as the average hop counts of all the substrate paths that virtual links in G_v have embedded to.

$$\overline{hops}(G_v)=\frac{\sum_{l_v\in L_v}hops(M^L(l_v))}{|L_v|} \tag{5}$$

where $M^L(l_v)$ denotes the path that l_v embed to.

The summarization of our works is illustrated in **Table 2**.

Table 2. Work summarization

Items	Contents
Metrics to optimize	The acceptance ratio
	The revenue to cost ratio
	Average Hop Counts of Link Embedding
The solving method	The NA-PVNM algorithm

4. Embedding Algorithm based on Node Connectivity Awareness and Path Integration Evaluation

In this section, we will introduce the details of our algorithm based on node connectivity awareness and path integration evaluation.

4.1 Node Embedding

In node embedding stage, we first sort virtual nodes in VN request. Because of the limited resources of substrate nodes, virtual nodes with more resource requirement are supposed to be priority embedded as their embedding face more difficult. Besides, the adjacency of virtual nodes is considered to sort virtual nodes by the BFS algorithm which can help reduce cost in link embedding stage.

We define function R to calculate resource requirement of virtual nodes.

$$R(n_v) = cpu(n_v) * \sum_{l_v \in L(n_v)} bw(l_v) \tag{6}$$

where $L(n_v)$ denotes the set of adjacent links of n_v.

The virtual nodes sorting process is as follows: calculate R of all virtual nodes, select virtual node with maximum value as the root node to run BFS algorithm; divide the rest virtual nodes into sets $\Omega^1_{n_v}, \Omega^2_{n_v}, \ldots, \Omega^n_{n_v}$ according to their distance to root node, where $\Omega^i_{n_v}$ represent the set in which nodes are i hop counts from root node; sort nodes in descending order in each set according to their R, and then we get the last embedding sequence of virtual nodes. In this way, each virtual node to be embedded in turn has an adjacent relationship with one or several virtual nodes that have been embedded, which can reduce path length of virtual links. **Fig. 2** shows an example of virtual network, and its virtual nodes embedding sequence is [4, 2, 1, 5, 6, 3, 7] according to the sorting method.

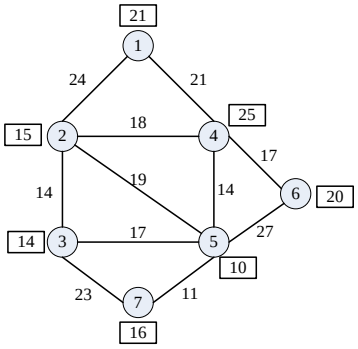


Fig. 2. An example of virtual network

Secondly, for each virtual node $n_{vi} \in N_v$ to be embedded, we develop a candidate node set with substrate nodes that satisfy resource and position constraint of n_{vi}, and denote it as $Can(n_{vi}) = \{n_s \,|\, dis(loc(n_{vi}), loc(n_s)) \le D(n_{vi}), cpu(n_{vi}) \le cpu(n_s)\}$.

Then we define function NF to calculate the fitness of n_s by considering its resource and connectivity.

$$NF(n_s) = \frac{R(n_s)}{Dis(n_s) + \varepsilon} \tag{7}$$

where $R(n_s)$ is defined in Eq. 6, $\varepsilon = 0.001$ is the parameter to prevent the dividend being zero. $Dis(n_s)$ is the connectivity parameter of n_s, it is calculated as follows: as n_{vi} is to be embedded,

we generate an available node set with substrate nodes which are embedded by the virtual nodes directly connected to n_{vi}. The set is denoted as $Embed(n_{vi}) = \{n_s \mid n_v \uparrow n_s, hops(n_v, n_{vi}) = 1\}$, where $n_v \uparrow n_s$ denote that n_v is embedded to n_s, $hops(n_v, n_{vi}) = 1$ denote that n_{vi} is directly connected to n_v in VN request. Then the connectivity parameter for each candidate substrate node is defined as

$$D(n_s) = \sum_{n_k \in Embed(n_{vi})} hops(n_s, n_k), n_s \in Can(n_{vi}) \tag{8}$$

Finally, n_{vi} is embedded to substrate node n_s that has maximum NF.

The details of our node-embedding algorithm are given by Algorithm 1.

Algorithm 1 Node Embedding Algorithm

Input: Substrate network G_s, Virtual network request G_v

Output: Node embedding M^N

1. **for** each virtual node $n_v \in N_v$
2. Calculate $R(n_v)$
3. **end for**
4. Take n_v with max R as root node, run BFS, divide the remaining nodes into sets $\Omega^1_{n_v}, \Omega^2_{n_v}, \ldots, \Omega^n_{n_v}$
5. Sort nodes in $\Omega^i_{n_v}$ in descending order according to their $R(n_v)$
6. Record the virtual nodes embedding sequence into *VirtualNodeList*
7. **for** each n_v in *VirtualNodeList* **do**
8. Generate candidate node set $Can(n_{vi})$
9. Update $Can(n_{vi})$ by $Can(n_{vi}) = Can(n_{vi}) \cap OpSubNode$
10. **if** $Can(n_{vi})$ is empty
11. **Return** NODE_EMBEDDING_FAILED
12. **else**
13. Generate the available node set $Embed(n_{vi})$
14. **for** each n_s in $Can(n_{vi})$
15. Calculate $NF(n_s)$
16. **end for**
17. Embed n_v to n_s with max NF, namely $M^N(n_v) = n_s$
18. Record n_s into set *OpSubNode*
19. **end if**
20. **end for**
21. **return** NODE_EMBEDDING_SUCCESS

4.2 Link Embedding

In link embedding stage, we use k-shortest path algorithm to calculate the k shortest paths between source and destination substrate nodes. Path is denoted as $p_k = (N_k, P_k)$, where N_k is the node set that p_k crosses, P_k is the link set that p_k crosses. We define function PF to calculate the fitness of each path, in which available bandwidth of path, available CPU of nodes across the path, and the length of path are considered.

$$PF(p_k) = \frac{\min\limits_{l_s \in P_k} bw(l_s)}{\max\limits_{n_s \in N_k} cpu(n_s)} * \frac{1}{hops(p_k)} \tag{9}$$

where $\min\limits_{l_s \in P_k} bw(l_s)$ is the minimum available bandwidth of substrate links in p_k, namely the path available bandwidth, $\max\limits_{n_s \in N_k} cpu(n_s)$ is the maximum available CPU of nodes in p_k, $\frac{\min\limits_{l_s \in P_k} bw(l_s)}{\max\limits_{n_s \in N_k} cpu(n_s)}$ can help balance resource consumption between nodes and links in SN, and reduce the possibility of network fragmentation, $hops(p_k)$ is the hop counts of p_k.

The details of our link-embedding algorithm are given by Algorithm 2.

Algorithm 2 Link-Embedding Algorithm

Input: Substrate network G_s ,Virtual network request G_v ,Node embedding M^N

Output: Link Embedding M^L

1. **for** each virtual link $l_{uv} \in L_v$ to be embedded **do**
2. Search the k-shortest paths between node u and v in substrate, record them to *Pathlist*
3. **if** *Pathlist* is empty **then**
4. **return** LINK_EMBEDDING_FAILED
5. **else**
6. **for** each $p_i \in Pathlist$
7. Calculate $PF(p_i)$
8. **end for**
9. Embed l_{uv} to p_i with max PF, namely $M^L(l_{uv}) = p$
10. **end if**
11. **end for**
12. **return** LINK_EMBEDDING_SUCCESS

4.3 Time Complexity Analysis

The time complexity of the algorithm includes: in node embedding stage, the time complexity of calculating the shortest path between substrate nodes is $O(|N_s|^2)$, to calculate NF for each candidate nodes of virtual node, the time complexity is $O(|N_v||N_s|^2)$, where $|N_s|$ depends on the position constraint. In link embedding stage, Yen algorithm [26] is used and its time complexity is $O(k|N_s|(|L_s|+|N_s|\log|N_s|))$. Thus the total time complexity of the algorithm is $O(|N_v||N_s|^2 + k|N_s|(|L_s|+|N_s|\log|N_s|))$. Therefore, NA-PVNM is a polynomial-time algorithm.

5. Performance Evaluation and Analysis

In this section, we first introduce the network settings in our simulation, then present the main evaluation results and analyze the performance of NA-PVNM in several aspects.

A simulator using Matlab to evaluate the performance of algorithms was developed. We generate SN and VN requests by using an improved Salam network topology generate algorithm. SN is composed by 100 nodes and about 500 links. The positions of substrate nodes

follow a uniform distribution in the scope of $L \times L = 1000 \times 1000$. The initial CPU of substrate nodes and bandwidth of substrate links are real numbers following a uniform distribution between 50 and 100. In SN, pairs of nodes are connected by links with the probability of

$$P = \beta_1 * e(\frac{-d^5}{\alpha_1 * L}) \tag{10}$$

where α_1 and β_1 are network characteristic parameter, α_1 determine the ratio between short links and long links, β_1 determine the number of links, d is the European distance between nodes, we set more short links in SN.

The number of nodes in each VN request is uniformly distributed between 5 and 15, and the average degree of each node is about 5. Positions of nodes follow a uniform distribution in the scope of $L \times L = 1000 \times 1000$ and all position constraints are set as 150. The required CPU and bandwidth resource are real numbers uniformly distributed between 10 and 30. The VN requests arrive by the Possion distribution with the rate of 10 per 100 time units. The lifetime of each VN request follows an exponentially distribution with the mean of 200 time units. Simulations were run 3000 time units to reach a stable state, which contain about 300 VN requests. To avoid the disturbance of random factors to the experimental results, each simulation is carried out for 10 times, and the average value was recorded as the final results.

The parameters that we use in our simulations are summarized in **Table 3**.

Table 3. Parameters in simulations

Parameters	Values
Number of substrate nodes	100
Positions of substrate nodes	a uniform distribution in the scope of $L \times L = 1000 \times 1000$
Number of substrate links	501
Substrate node CPU and BW resources	A uniform distribution from 50 to 100
Number of virtual nodes	A uniform distribution from 5 to 15
Positions of virtual nodes	a uniform distribution in the scope of $L \times L = 1000 \times 1000$
Position constraints of virtual nodes	$D(n_{vi}) = 150$
Virtual node CPU and BW requirements	A uniform distribution from10 to 30
Arrival rate of VN requests	10 per 100 time units
Lifetime of VN request	200 time units exponentially distribution

Our simulation include four parts, first, the performance of proposed algorithm is verified in comparison with 3 crucial previous algorithms; second, impact of position constraint on the performance of algorithm is analyzed via simulation; third, impact of substrate network topology attributes on the performance of algorithm is analyzed via simulation; fourth, the utilization of substrate network under different network setting is analyzed.

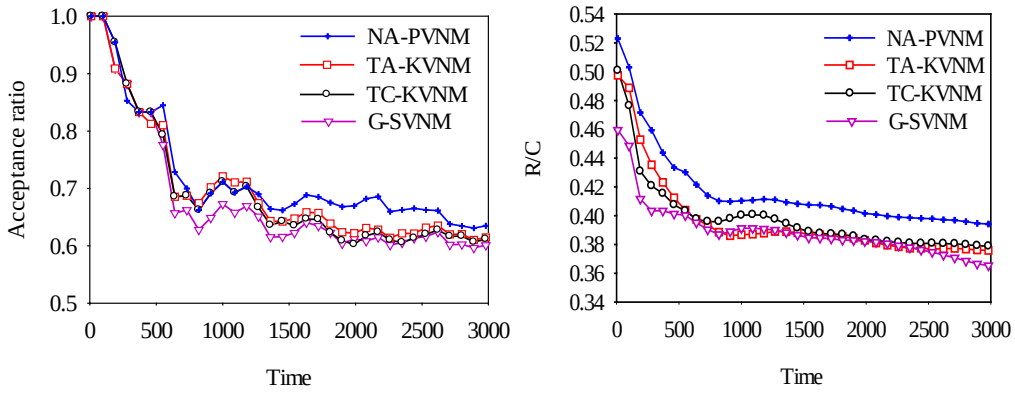
5.1 Performance Comparisons

The metrics to evaluate VN embedding algorithms are acceptance ratio of VN requests and R/C (Eqs. (1, 4)). Our simulations focus on four algorithms that list in **Table 4.** NA-PVNM is the algorithm we proposed, TA-KVNM [12], and TC-KVNM [9] consider topology attributes of substrate and virtual networks for VN embedding, G-SVNM is the greedy algorithm.

Table 4. Algorithms comparison

Notation	Description
NA-PVNM	The proposed algorithm with node embedding stage that sort virtual nodes by R and BFS algorithm, select substrate node with maximum NF ,link embedding stage that use k-shortest path algorithm and select path with maximum PF .
TA-KVNM	Algorithm with node embedding stage that rank virtual and substrate nodes by TOPSIS method considering CPU, adjacent bandwidth and closeness centrality of nodes, link embedding stage that use k-shortest path algorithm.
TC-KVNM	Algorithm with node embedding stage that sort virtual nodes by R , select substrate node with maximum NF , link embedding stage that use k-shortest path algorithm.
G-SVNM	Embedding nodes with greedy local resource and links with the shortest path algorithm.

The comparison results are shown in **Fig. 3.** **Fig. 3-a** illustrates the acceptance ratio of four algorithms. The acceptance ratio falls down at beginning, and reaches the stable state (about 0.68) at about $t = 800$ until the end. This is because resources of substrate network are richness at beginning, then the consumption of resource leads to the bottleneck of embedding, and with the resource released due to the ending of VN requests, available resource of substrate network maintain stable. The acceptance ratio of NA-PVNM(0.66 on stable) is better than the other algorithms, at 5% higher than G-SVNM(0.61 on stable), 4% higher than TA-KVNM and TC-KVNM(0.62 on stable). **Fig. 3-b** illustrates R/C of four algorithms, the R/C of NA-PVNM (0.4 on stable) is better than the other algorithms, at 3% higher than G-SVNM(0.37 on stable), 2% higher than TA-KVNM and TC-KVNM(0.38 on stable). Simulation results show that our new algorithm has better performance than others, and the optimization strategy adopted in our algorithm is effective.



(a) Acceptance ratio (b) R/C

Fig. 3. Comparison between our algorithm and others

What can be seen in **Fig. 3** is that the performance of algorithms is approximate. Factors including the scale and resource richness of substrate network, the scale and resource requirement of VN requests, arrival rate and survival time of VN requests do have a significant impact on the acceptance ratios, but have little impact on the performance difference. Excluding those factors, we conclude that the reasons lead to the approximate performance are as follows: the setting of position constraint is small which lead to the limited choice space of node embedding; the substrate network topology in our simulation is uniformity, thus the hop counts of different paths between substrate node pairs are similar under the position constraint,

which may lead to the similar results of link embedding. The two factors together restrict the solution space of algorithms, which make the performance approximate. Thus, we design the following experiments to analyze the impact of position constraint and substrate network topology attributes on the performance of algorithms.

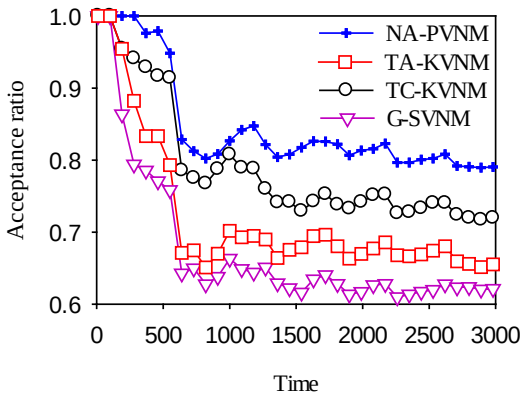
5.2 Evaluation with different position constraint

To evaluate the impact of different position constraint on the performance of algorithms, the attributes of VN requests and SN are the same as those in the network settings, and would not change; we set the position constraint of all virtual nodes as $D(n_{vi})=400$ and $D(n_{vi})=1000$ respectively. The metrics to evaluate the impact of position constraint on algorithm performance are as follows: the acceptance ratio, R/C, and the average hop counts of link embedding (Eqs. (1, 4, 5)).

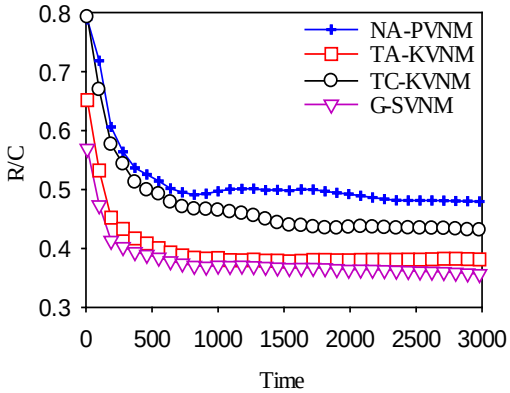
Figs. 4-a and **-b** illustrate the acceptance ratio and R/C of algorithms under $D(n_{vi})=400$. As we can see, the acceptance ratio of NA-PVNM(0.8 on stable) is better than others, at 5% higher than TC-KVNM (0.75 on stable), 14% higher than TA-KVNM(0.65 on stable) and 18% higher than G-SVNM (0.62 on stable), what's more, acceptance ratio of NA-PVNM and TC-KVNM under $D(n_{vi})=400$ improves about 14% and 13% than those under $D(n_{vi})=150$. R/C of NA-PVNM(0.5 on stable) is better than others, at 5% higher than TC-KVNM (0.45 on stable), 11% higher than TA-KVNM(0.39 on stable) and 12% higher than G-SVNM (0.38 on stable), what same is that R/C of NA-PVNM and TC-KVNM under $D(n_{vi})=400$ improves than those under $D(n_{vi})=150$, while TA-KVNM and G-SVNM go through without those improving.

Figs. 4-c and **-d** illustrate the acceptance ratio and R/C of algorithms under $D(n_{vi})=1000$. As we can see, the performance of NA-PVNM and TC-KVNM improves along with $D(n_{vi})$'s widen, they are better than those under $D(n_{vi})=400$, acceptance ratio of NA-PVNM(0.97 on stable) and TC-KVNM(0.87 on stable) improved 17% and 12% respectively, R/C of NA-PVNM(0.75 on stable) and TC-KVNM(0.67 on stable) improved 25% and 22% respectively, while TA-KVNM and G-SVNM go through without improving.

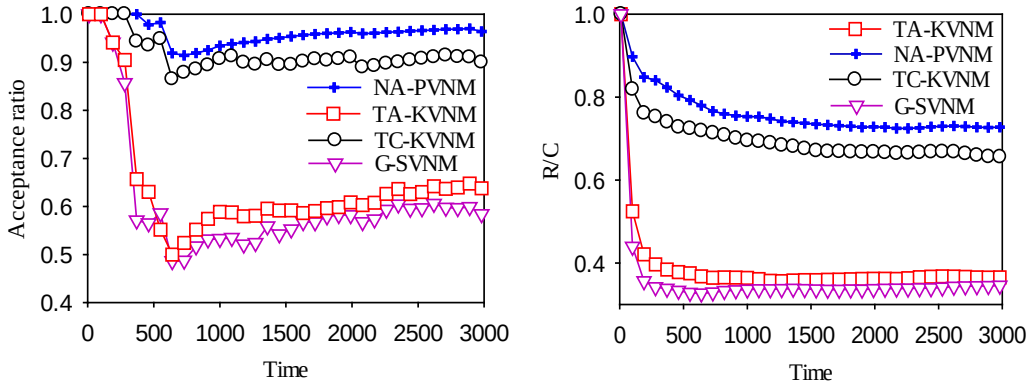
The simulation results show that NA-PVNM has better performance, and the position constraint has a significant impact on the performance of algorithms. With the enlargement of position constraint, the performance of NA-PVNM and TC-KVNM are greatly improved, but TA-SVNM and G-SVNM have little change in performance.



(a)Acceptance ratio under $D(n_{vi})=400$



(b) R/C under $D(n_{vi})=400$



(a)Acceptance ratio under $D(n_{vi})=1000$ (b) R/C under $D(n_{vi})=1000$

Fig. 4. Comparison between our algorithms and others under different position constraint

Table 5 shows the average hop counts of four algorithms. As we can see, $\overline{hops}$ of NA-PVNM and TC-KVNM decrease, while $\overline{hops}$ of TA-KVNM and G-SVNM has little change with the increase of $D(n_{vi})$. This is because NA-PVNM and TC-KVNM select closer substrate nodes in node embedding stage, and reduce the path length of link embedding, thus the consumption of bandwidth resource reduces, which squeeze more room out for following VN requests. At the same time, the less cost of VN request will increase the R/C. As a comparison, the node embedding stage of TA-SVNM and G-SVNM does not take the distance factor into consideration: the closeness centrality that TA-SVNM considers cannot directly reflect the connectivity between nodes; the $\overline{hops}$ of G-SVNM increases after enlarging the node position constraint as its node embedding stage is more blindly.

Table 5. Average hop counts of algorithms under different position constraint

	NA-PVNM	TA-KVNM	TC-KVNM	G-SVNM
$D(n_{vi})=150$	3.2735	3.3690	3.3287	3.4565
$D(n_{vi})=400$	2.5617	3.2963	2.7699	3.5741
$D(n_{vi})=1000$	1.3590	3.4730	1.5060	3.9085

Results of the evaluation show that the enlargement of node position constraint helps reduce path length of link embedding, improve the acceptance ratio and R/C of VN embedding.

5.3 Evaluation with different substrate network attributes

The metrics to evaluate the impact on the algorithm performance with different substrate network attributes are as follows: the acceptance ratio, R/C, and the average hop counts of link embedding **(Eqs. (1, 4, 5))**.

We set more short links than long links in SN setting, as a result, the larger European distance between substrate node pairs, the longer path length between the node pairs. In this regard, we change the substrate network topology attributes by adjusting the parameter α、β in Eq. (10), and describe topology attributes by network diameter which is denoted as $NetD$, and average shortest path length which is denoted as $NetL$. We define them as

$$NetD = \max_{i,j} d_{ij} \tag{11}$$

$$NetL = \frac{1}{|N_s|(|N_s|-1)} \sum_{i \neq j} d_{ij} \tag{12}$$

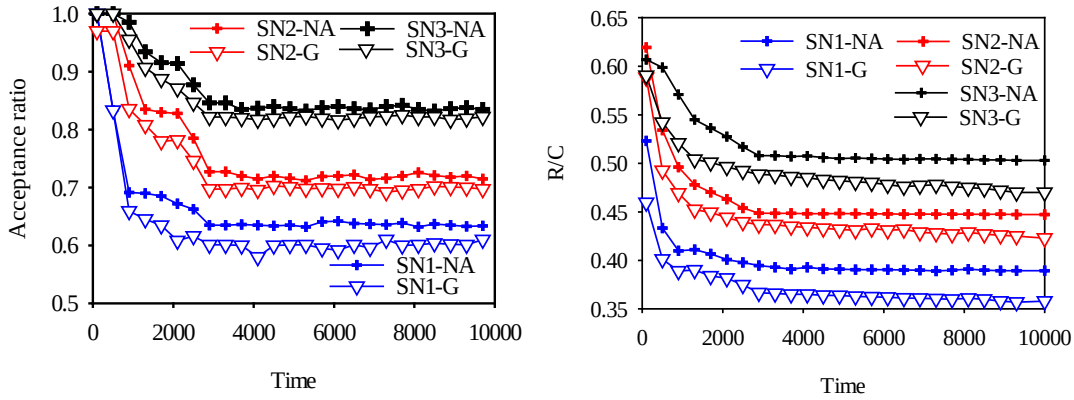
where d_{ij} is the shortest path length between node i and j.

The substrate network generating in Section 5.1 is denoted as Subnet1, and as a comparison, we generate two new substrate networks. **Table 6** shows the attributes of substrate networks. As we can see, network attributes including the number of substrate nodes and links, the total resource of node CPU and link bandwidth are similar; topology attributes of *NetD* and *NetL* are different. There are most short links and least long links setting in Subnet1, which result in the least *NetD* and *NetL*. As a comparison, there are most long links and least short links setting in Subnet3, which result in the maximum *NetD* and *NetL*.

Table 6. Attributes of substrate networks

	$\|N_s\|$	$\sum_{n_s \in N_s} cpu(n_s)$	$\|L_s\|$	$\sum_{l_s \in L_s} bw(l_s)$	*NetD*	*NetL*
Subnet1	100	7581	501	37155	8	3.4844
Subnet2	100	7481	510	38470	6	2.9293
Subnet3	100	7563	509	38005	4	2.2149

Other parameters are the same as those in Section 5.1, simulations were run 10000 time units to reach a stable state, and we evaluate the impact of substrate network attributes on algorithm performance by simulating NA-PVNM and G-SVNM on the substrate networks described in **Table 3**, the acceptance ratio and R/C are shown in **Fig. 5**. Firstly, as we can see, as the average shortest path length of substrate network decrease, the acceptance ratio and R/C of two algorithms increase. **Fig. 5 (a)** illustrates that the acceptance ratio of NA-PVNM(0.85 on stable) and G-SVNM(0.82 on stable) on Subnet1 are about 12% higher than those on Subnet2(0.73 and 0.7 on stable), about 20% higher than those on Subnet1(0.65 and 0.6 on stable). **Fig. 5 (b)** illustrates that the R/C of NA-PVNM(0.53 on stable) and G-SVNM(0.5 on stable) on Subnet3 are about 8% higher than those on Subnet2(0.46 and 0.45 on stable), about 12% higher than those on Subnet1(0.41 and 0.37 on stable). On the other hand, we can see that two algorithms have similar performance on different SN, this is because the position constraint setting is small.



(a) Acceptance ratio (b) R/C

Fig. 5. Comparison between NA-PVNM and G-SVNM on different substrate network

Table 7 shows the comparison of the average shortest path length of substrate networks, and the average hop counts of two algorithms on three substrate networks. It can be seen that the average shortest path length has a significant impact on the average hop counts of link embedding. As the average shortest path length decreases, length of link embedding decreases, thus the resource consumption of bandwidth resource reduce, as a result, the acceptance ratio increases, and R/C increases as the less cost of VN request.

Table 7. Comparison between substrate networks' average shortest path length and $\overline{hops}$ of algorithms

	NetL	$\overline{hops}$ of NA-PVNM	$\overline{hops}$ of G-SVNM
Subnet1	3.4844	3.2735	3.4565
Subnet2	2.9293	2.8149	2.977
Subnet3	2.2149	2.2293	2.4951

Results of evaluation show that, increase the proportion of long links in substrate network will help reduce the hop counts of link embedding, and improve the performance of VN embedding.

5.4 Evaluation of substrate network resources usage

Results of evaluations in section 5.2 and 5.3 show that the enlargement of position constraint and increasing the proportion of long links in substrate network can help improve the performance of virtual network embedding. Their common ground is to reduce the hop counts of link embedding, and as a result, the resource consumption of substrate network is reduced, the acceptance ratio and R/C are improved. Based on this, we design the following experiment to analysis the resource utilization of substrate network during embedding under different network settings, the factors affecting the performance of algorithms are analyzed as well.

The metrics we use to evaluate the resource utilization of substrate network include the ratio between substrate links whose load are more than 90% with the total substrate links, the time average of ratio between occupied bandwidth with total bandwidth of links, we denote them as $link\ ratio$ and $bw\ ratio$ respectively, and define the two ratios as:

$$link\ ratio = \frac{\sum l_{s\_0.9}}{|L_s|} \tag{13}$$

$$bw\ ratio = \frac{\sum_{t \to T} \frac{\sum_{l_s \in L_S} Tbw(l_s) - bw(l_s, t)}{\sum_{l_s \in L_S} Tbw(l_s)}}{t} \tag{14}$$

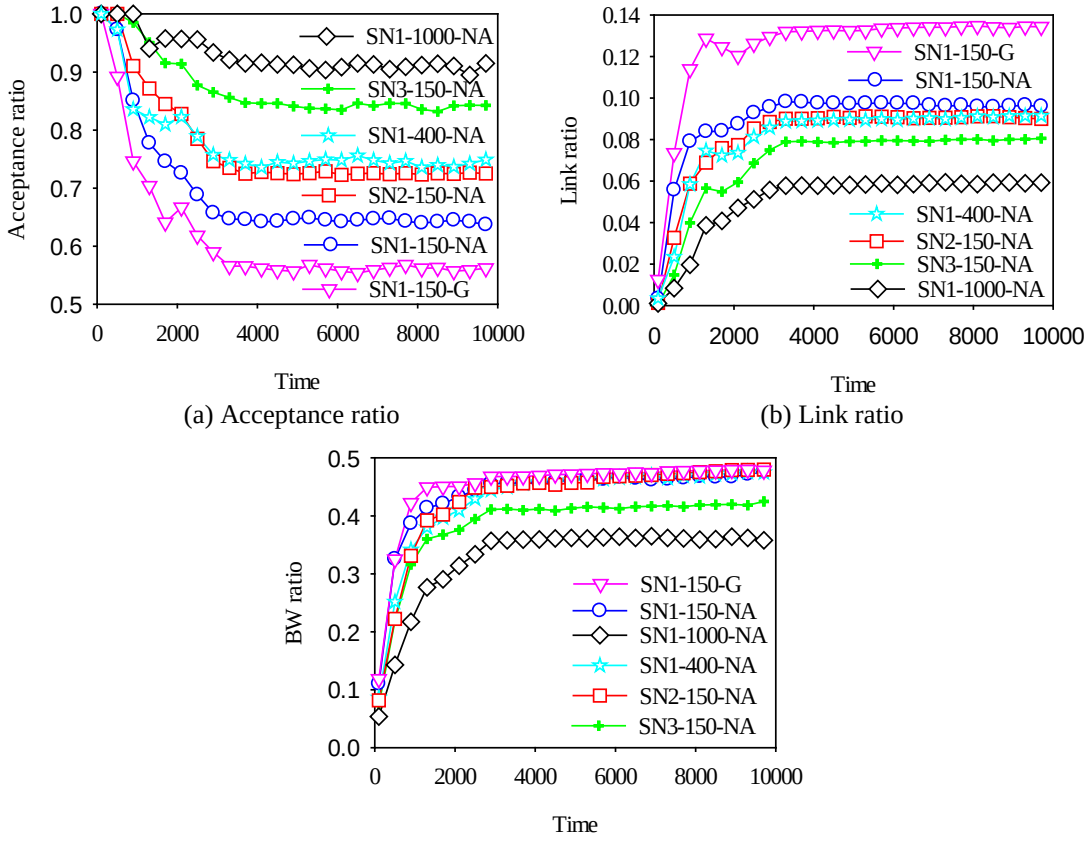
where $\sum l_{s\_0.9}$ is the number of substrate links whose load are more than 90%, $Tbw(l_s)$ is the total bandwidth of l_s, $bw(l_s, t)$ is the available bandwidth of l_s at time t.

The $link\ ratio$ is defined by considering that the minimum bandwidth requirement of virtual links in VN requests is set as 10, and the maximum bandwidth of substrate links is set as 100, so that substrate links whose load are more than 90% will become the bottleneck and partition substrate network. The $bw\ ratio$ reflects the overall occupancy of substrate network resource.

We set the acceptance ratio of G-SVNM on Subnet1 with $D(n_{vi}) = 150$ as a baseline, put the acceptance ratio of NA-PVNM on different substrate networks under $D(n_{vi}) = 150$, the acceptance ratio of NA-PVNM on Subnet1 under different $D(n_{vi})$ together, and illustrate the comparison in **Fig. 6 (a)**. The comparison of $link\ ratio$ and $bw\ ratio$ are illustrated in **Figs. 6**

(b) and **(c)**. It can be seen in **Figs. 6 (a)** and **(b)** that the acceptance ratio is significantly associated with *link ratio*, they reach to the stable state at similar time. Besides, the higher *link ratio* is, the least acceptance ratio is. As we can see, SN1-150-G setting results in the least acceptance ratio and the maximum *link ratio*, SN1-1000-NA setting results in the maximum acceptance ratio and the least *link ratio*.

It can be seen in **Figs. 6 (a)** and **(c)** that for network settings (SN1-150-G, SN1-150-NA, SN1-400-NA, SN2-150-NA) whose acceptance ratio is relative low(less than 0.75 on stable), their *bw ratio* is similar. The result shows that as the acceptance ratio reaches a stable state, the resource utilizations of substrate network are close for different network settings. The acceptance ratio of SN1-400-NA and SN2-150-NA is higher than those of SN1-150-G and SN1-150-NA, this is because the VN embedding of SN1-400-NA and SN2-150-NA consumes less bandwidth resources(as described in **Table 5** and **7**, which squeeze more room out for following VN requests embedding. SN1-1000-NA has the least *link ratio* and the maximum acceptance ratio, because the resource consumption for each virtual network is least under the setting.



(a) Acceptance ratio (b) Link ratio

(c) Bw ratio

Fig. 6. Comparison of algorithm under different network setting

The main conclusions of the results are as follow: under the constraint of substrate network resource distribution and virtual network position constraint, as the utilization of resource reaches a certain level, there will be bottleneck in substrate network which results in the falling down of VN acceptance ratio or reaching a stable state, at the same time, the key factor to improve the acceptance ratio of VN requests is to reduce the resource consumption of VN

embedding.

6. Conclusions

In this paper, we study the VN embedding problem from the perspective of considering the virtual node position constraint and substrate network topology attributes. We propose a two -stage virtual network embedding algorithm. Simulation shows that the proposed algorithm reduces the path length of the virtual link embedding, increases the acceptance ratio and the ratio between revenue and cost of virtual network requests, and improves the resource utilization of substrate network. We analysis the impact of virtual node position constraint and substrate network topology attributes on the performance VN embedding, and the resource usage of substrate network during the embedding process. by the experimental conclusion, under the constraints of substrate network resources distribution and the virtual network position constraint, as the usage of substrate network resources reaches a certain level, there will be bottleneck in substrate network which result in the falling down of VN acceptance ratio or reaching a stable state, at the same time, the key factor to improve the success rate of virtual network embedding is to reduce the resources consumption of VN embedding.

However, there are also some problems appeared during our experiments. Firstly, the nicer performance and conclusion are under restrictions: SN and VN generated and used are random networks which are different from actual networks in some characteristics; parameters setting in our simulation results in the condition that bandwidth resources become the bottleneck of successful embedding. Secondly, indicators are integrated in a simple but inflexible way in NA-PVNM, dimension of indicators are different that restrict how to use them. In the future work, we will extend our research in three aspects: firstly, more types of SN and VN will be generated, scale-free networks, small world networks and some actual networks will be used as SN, together with different scale and regular VN will be used to explore the VN embedding process. Secondly, more strategies, how these strategies affect the embedding process and the integration of strategies with more flexible and extendable are needed to try. Thirdly, network diameter and average shortest path length are primarily used to evaluate the effects, network attributes of betweenness centrality, degree distribution, clustering coefficient and so on, and different values of them in different type network are needed to explorer. Above all, we believe that our work will have guiding significance to the future research.

References

[1] A. Wang, M. Iyen, R. Dutta and et al., “Network virtualization: technologies, perspectives, and frontiers,” *Journal of Lightwave Technology*, vol.31, no.4, pp.523-537, August, 2012. Article (CrossRef Link)

[2] T. Anderson, L. Peterson, S. Shenker and et al., “Overcoming the Internet impasse through virtualization,” *Computer*, vol.38, no.4, pp. 34-41, April, 2005. Article (CrossRef Link)

[3] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer and X. Hesselbach, “Virtual network embedding: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888-1906, February, 2013. Article (CrossRef Link)

[4] X. Cheng, Z. Zhang, S. Su and et al., “Survey of virtual network embedding problem,” *Journal on Communications*, vol.32, no.10, pp 143-151, October, 2011. Article (CrossRef Link)

[5] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. of 25th IEEE INFOCOM Conference*, Loc: Barcelona, Spain, pp. 1-12, April 23-29, 2006. Article (CrossRef Link)

[6] M. Yu, Y. Yi and J. Rexford, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol.38, no.2, pp. 17-29, April, 2008. Article (CrossRef Link)

[7] M. Chowdhury, M.R. Rahman and R. Boutaba, "Vineyard: virtual network embedding algorithms with coordinated node and link embedding," *IEEE/ACM Trans. Net*, vol.20, no.1, pp.206–219, July, 2012. Article (CrossRef Link)

[8] X. Cheng, S. Su and Z. Zhang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38-47, April, 2011. Article (CrossRef Link)

[9] M. Feng, L. Zhang and X. Zhu, "Topology-aware virtual network embedding through the degree," in *Proc. of National Doctoral Academic Forum on Information and Communications Technology*, Beijing, China, pp.1-6, August 21-23, 2013. Article (CrossRef Link)

[10] H. Cui, W. Gao and J. Liu, "A virtual network embedding algorithm based on virtual topology connection feature," in *Proc. of IEEE Wireless Personal Multimedia Communications(WPM)*, Atlantic City, USA, pp. 1-5, June 24-27, 2013. Article (CrossRef Link)

[11] Z. Wang, Y. Han, T. Lin and et al., "Topology-aware virtual network embedding based on closeness centrality," *Frontiers of Computer Science in China*, vol. 7, no. 3, pp. 446–457, February, 2013. Article (CrossRef Link)

[12] J. Ding, T. Huang, J. Liu and Y. Liu, "Virtual network embedding based on real-time topological attributes," *Frontiers of Information Technology & Electronic Engineering*, vol.16, no.2, pp.109-118, February, 2015. Article (CrossRef Link)

[13] S. Nashid, A. Reaz, R. Shihabur and et al., "Connectivity-aware Virtual Network Embedding," in *Proc. of 2016 IFIP Networking Conference and Workshops*, Vienna, Austria, pp.45-54, May 17-19, 2016. Article (CrossRef Link)

[14] L. Peng, "Virtual network embedding algorithm based on breadth-first search," *Journal of Sichuan university (engineering science edition)*, vol. 47, no.2, pp. 117-122, March, 2015. Article (CrossRef Link)

[15] J. Liao, M. Feng, T. Li, J. Wang and S. Qing, "Topology-aware virtual network embedding using multiple characteristics," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 1, pp. 145-164, January, 2014. Article (CrossRef Link)

[16] N. Qi, B. Wang, B. Wang and et al., "Research on Balanced Construction Algorithm of Virtual Network," *Journal of Electronics & Information Technology*, vol. 33, no. 6, pp. 1301-1306, October, 2011. Article (CrossRef Link)

[17] X. Li, H. Lu, W. Zhou and P. Hong, "VNE-RFD: Virtual Network Embedding with Resource Fragmentation Consideration," in *Proc. of 2014 IEEE Global Communications Conference*, Austin, USA, pp.1842-1847, December 7, 2014. Article (CrossRef Link)

[18] S. Gong, J. Chen, C. Huang, and Q. Zhu., "Trust-aware secure virtual network embedding algorithm," *Journal on Communications*, vol.36, no.11, pp.180-189, November, 2015. Article (CrossRef Link)

[19] G. Liu and S. Su, "The research of reliable virtual network embedding algorithm," *Acta Electronica Sinica*, vol.44, no.8, pp.1820-1825, 2016. Article (CrossRef Link)

[20] D. Liao, G. Sun, V. Anand and H, Yu, "Efficient provisioning for multicast virtual network under single regional failure in cloud-based datacenters," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 7, pp. 2325-2349, 2014. Article (CrossRef Link)

[21] S. Nashikd, A. Reaz and R. Shihabur, "Connectivity-aware Virtual Network Embedding," in *Proc. of 2016 IFIP Networking Conference and Workshops*, Vienna, Austria, pp.46-54, May, 17-19, 2016. Article (CrossRef Link)

[22] M. Beck, P. Linnhoff and A. Fischer, "A simulation framework for Virtual Network Embedding algorithms," in *Proc. of Proceedings of the IEEE Telecommunications Network Strategy and Planning Symposium*, Funchal, Portugal, pp.1-6, September 17-19, 2014. Article (CrossRef Link)

[23] J. Ding, T. Huang, J. Wang and W. Hu, "Virtual network embedding through node connectivity," *The Journal of China Universities of Posts and Telecommunications*, vol.22, no.1, pp.17–23, March, 2015. Article (CrossRef Link)

[24] B. Huang, R. Lin, K. Peng, H. Zou and F. Yang, "Load-balancing based on particle swarm optimization in virtual network embedding," *Journal of Electronics & Information Technology*, vol.35, no.7, pp. 1753-1759,Julu, 2013. Article (CrossRef Link)

[25] Z. Zhang, S. Su, Y. Lin, X. Cheng and K. Shuang, "Adaptive multi-objective artificial immune system based virtual network embedding," *Journal of Network and Computer Applications*, vol.53, no.1, pp. 140–155, July, 2015. Article (CrossRef Link)

[26] D. Satria, D. Park and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, no. 1, pp. 138–147, May, 2017. Article (CrossRef Link)

[27] J. YEN, "Finding the K shortest loop less paths in a network," *Management Science*, vol.17, no.11, pp. 712-716, 1971. Article (CrossRef Link)

Zhiyuan Zhao is a Ph.D. candidate in Computer Application Technology from Air Force Engineering University, China. His research interests include virtual network embedding and software defined networking.

Xiangru Meng is a professor in Communication and Information System from Air Force Engineering University, China. His research interests include next generation Internet, cloud computing and software defined networking.

Yuze Su is a Ph.D. candidate in Computer Application Technology from Air Force Engineering University, China. His research interests include network virtualization and cloud computing.

Zhentao Li is a postgraduate in Computer Science from Air Force Engineering University, China. His research interests include network virtualization and network survivability.